**Lab 3 – SCANNER**

**REQUIREMENTS**

Implement a scanner (lexical analyzer): Implement the scanning algorithm and use ST from lab 2 for the symbol table.

Input: Programs p1/p2/p3/perr and token.in (see Lab 1a)

Output: PIF.out, ST.out, message “lexically correct” or “lexical error + location”

Deliverables: input, output, source code, documentation

**DOCUMENTATION**

For both symbol tables of identifiers and constants I used the ST implemented in the previous lab.

For the program information file, I am using a list of pairs: code + position in ST (both unique). If the element is not in the ST (operators, separators, reserved words) position is (-1, -1).

The codes are provided in the tokens.in file, from where they are collected at the beginning of the program (identifiers: 0, constants: 1).

The program starts reading line by line the file it was provided, and tokenizes it on the spot – it divides the line in a list of each element on the line: identifiers, constants, operators, separators. At this point, the program starts identifying each element from the tokenized line: if the codes list contains the token, it only adds it to the PIF as it is a reserved word, separator or operator. If not, then it adds it to the corresponding symbol table (STI – identifiers or STC – constants) and then to the PIF. If there is any token that cannot pe identified, an error is returned along with the line it occurs on.

**CLASS SCANNER**

public List<String> run(Path path)

* returns the list of errors which occurred while scanning the file provided in the path

private ArrayList<String> tokenizeLine(ArrayList<String> tokens, String separator)

* tokenizes the line provided and returns the list of obtained tokens

private void identify(String token) throws IllegalNameException

* categorizes the token as separator/operator/reserved word, identifier or constant

private void addToPIF(String token)

* adds the corresponding pair of code – position to the PIF

private void addToST(String token, SymbolTable ST)

* adds the token to its position in the corresponding symbol table

private void getCodes()

* retrieves the codes from the file tokens.in

